Practical Assignment Report: Human Activity Recognition using Smartphones

Connectivity and Pattern Recognition Course 2016/2017

Ricardo da Silva Carvalho Mendes

June 5, 2017

1 Introduction

This document reports the work done for the practical assignment "Human Activity Recognition using Smartphones" for the connectivity and pattern recognition 2016/2017 course. This work consisted in the development of efficient classifiers for human activity recognition, where two scenarios were considered:

i) Binary problem – to determine if a person is walking or not walking;

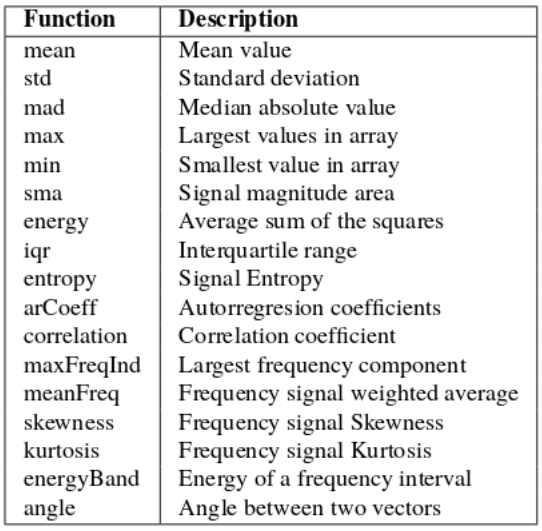
ii) Multiclass problem – to differentiate between six different activity states (walking, walking upstairs, walking downstairs, sitting, standing and laying)

The dataset used in this work can be found online in kaggle's repository[1], and contains all the required data. Using this dataset, the following steps were taken to tackle the aforementioned problems: data preprocessing; feature extraction; feature selection; classifiers training and evaluation. The remainder of this report will detail each of these steps by the following order. Section 2 describes the dataset and the data preprocessing step. Section 3 present the techniques used in this work for feature extraction and feature selection. The implemented classifiers are described on Section 4. Section 5 details the experimental procedure, and presents and analyzes the obtained results. Section 6 concludes this work.

2 Dataset and Preprocessing

As aforementioned, the dataset was obtained from kaggle's repository (`https://www.kaggle.com/uciml/human-activity-recognition-with-smartphones`). It was collected from 30 participants within the ages of 19 to 48 years, performing daily living activities such as walking and sitting. Each person wore a smartphone from which the accelerometer's and gyroscope's 3-axial signals (linear acceleration and angular acceleration, respectively) were captured at a 50Hz constant rate. The smartphones' signals were pre-processed [1] by applying noise filters and then sampled in fixed-width sliding windows of 2.56 sec and 50% overlap (128 readings/window). The sensor acceleration signal, which has gravitational and body motion components, was separated using a Butterworth low-pass filter into body acceleration and gravity. The gravitational force is assumed to have only low frequency components, therefore a filter with 0.3 Hz cut-off frequency was used. From each window, a vector of **561**-features was computed

[1] `https://www.kaggle.com/uciml/human-activity-recognition-with-smartphones`



Function	Description
mean	Mean value
std	Standard deviation
mad	Median absolute value
max	Largest values in array
min	Smallest value in array
sma	Signal magnitude area
energy	Average sum of the squares
iqr	Interquartile range
entropy	Signal Entropy
arCoeff	Autorregresion coefficients
correlation	Correlation coefficient
maxFreqInd	Largest frequency component
meanFreq	Frequency signal weighted average
skewness	Frequency signal Skewness
kurtosis	Frequency signal Kurtosis
energyBand	Energy of a frequency interval
angle	Angle between two vectors

Figure 1: List of measures for computing feature vectors. Source: [1].

by calculating variables from the time and frequency domain. Figure 1 presents the full list of these features.

The dataset presented no missing values, nor apparent noise/outliers, thus no data cleaning was required. However the data was not normalized. Therefore, the **z-score normalization** [2] was applied to the dataset.

A total of 10299 records (points) were in the dataset, partitioned in 70% for training data and 30% for test data. In this work, this partition was not used, and thus, the training and test data were joined together to allow for other cross-validation techniques.

	LAYING	SITTING	STANDING	WALKING	WALKING DOWN-STAIRS	WALKING UPSTAIRS
nr. elements (%)	∼18.90	∼17.25	∼18.50	∼16.75	∼13.60	∼15.00

Table 1: Approximate class distribution in terms of percentage for both training and testing. Percentages can slightly very due to the use of k-folds cross validation (k=10 folds).

Table 1 presents the percentual distribution of the different classes in the dataset. Due to the use of cross-validation, which randomly select the elements, these percentages slightly vary between the training and the test dataset. It can be seen that the dataset is highly unbalanced for both the binary problem and for the multiclass problem. For the binary case, only the class WALKING was used (positive), where all other elements were considered "NON-WALKING" (negatives), thus resulting in a 16.75% of positives and 83.25% of negatives. For the multiclass problem and in one-against-all approaches (explained in section 4), the most unbalanced case is 13.60% of positives belonging to the class WALKING_DOWNSTAIRS and 86.4% negatives. In one-versus-one approaches (explained in section 4), the worst case scenario is LAYING versus WALKING_DOWNSTAIRS with an approximate partition of 72% for LAYING points versus 28% of WALKING_DOWNSTAIRS. This data skewness can difficult the learning task of classifiers [3], since the class of interest (positive) is the one with less instances.

There are several ways to handle unbalanced datasets [3]. In this work, **random under-sampling** was used to randomly remove points from the classes as to match the number of elements of the class with less points (minority class). For this dataset, the minority class corresponded to WALKING_DOWNSTAIRS (1406 points), and with this operation, the dataset was reduced by 18.09% (to 8436 points, from the original 10299). According to [4], a good practice for training classifiers is to have at least ten times as many number of training samples per class n as the number of features d ($n/d > 10$). In this work, this ratio was respected for both the single and multiclass problem.

3 Feature Extraction and Selection

Feature extraction and feature selection are methods to reduce the dimensionality of the dataset [4]. While this may reduce the feature discrimination power, it reduces the learning complexity, thus allowing for faster training phases and lighter (w.r.t. memory space) classifiers.

In this work, two feature extraction methods have been used: principal component analysis (pca) and locally linear embedding (lle). Both these approaches are unsupervised, and thus do not take into consideration the true labels. However, while pca is a linear feature extraction method, lle can produce non-linear features.

In this work, the used pca implementation is from the Statistical Pattern Recognition Toolbox (SPRT) [5]. This implementation transforms the data such that the reconstruction error is minimized. The percentage of variance to retain can be passed as input. A visual inspection on the eigenvalues can be done to select how many eigenvalues to use, which in turn define the final dimension of the feature vectors. The lle implementation used in this work can be found in `http://cs.nyu.edu/~roweis/lle/code.html`, and it accepts as input parameter the number of max dimensions to retain.

For feature selection, the feature correlation, feature to class correlation and area under the curve (auc) values were analyzed. The first of these methods consists in measuring the correlation between the features, and select the features that are less correlated. Less correlated features means that each feature is more distinct from the others. In the feature to class correlation, the correlation between each pair class-feature is analyzed. In this context, a higher correlation is indicative that the feature is a good class indicator, and thus should be preserved. The auc values can be computed by using the true class labels as targets and the features as predicted values to build the ROC curve. In this sense, the features with higher auc suggest higher discriminant power. For all these three feature selection methods, a threshold has to be defined as to select the more discriminant features.

4 Classification and Evaluation

For this work, three different classifiers were used: the k-nearest neighbor (k-nn), a support vector machine (SVM) and the k-means. Each of these classifiers is different and require different input parameters. Bellow is presented a small explanation about each of the classifiers, how the parameters were set/adjusted and how the results were evaluated for both the single class and multiclass problem.

The implementation of the k-nearest neighbor used in this work is the k-nn provided in the Statistical Pattern Recognition Toolbox (SPRT) [5]. A k-nn classifier essentially searches for the k nearest points (neighbors) from the point one seeks to classify, and selects the class to which the majority of the neighbors belong. The implementation from SPRT is limited to the use of the euclidean distance to find the k nearest neighbors. Thus, the only parameter left to set is k. An empirical rule-of-thumb to select k is to choose $k = \sqrt{n}$ [6], and preferably odd to avoid draws. While good results were obtained using this value, similar results where obtained using $k = 1$, for both problems (see section 5). Since setting $k = 1$ is a simpler and faster approach, this value was chosen as the parameter.

Support vector machines (SVMs) are traditionally binary classifiers, that find a linear hyper-plane that maximizes the distance from this hyper-plane to the closest positive class and negative class points. SVMs have been extended to non-linear and multiclass classification. The SVM used in this work was the C-support vector classification (C-SVC) from the LIBSVM library [7], which can be used for single class and multiclass problems. This particular implementation requires, as input parameters, the kernel type, which I have selected the radial basis function (RBF), a factor γ for the kernel and C, the regularization parameter (cost). A practical approach to find (tune) γ and C is to perform a grid-search using cross-validation [8]. This process consists in training the classifier for various pairs (C,γ), as a discrete grid, and to evaluate the performance of each classifier. The best classifier has the best combination of (C,γ) from the ones that were tried.

k-means clustering [9] attempts to form a partition of the data in k clusters such that a criterion is met. k-means is an unsupervised approach, and thus, a training and a test phases are not well defined. One can use the training dataset to form the partition of the feature space in k clusters, and evaluate the results with the test data considering each cluster as a class. Thus, k was set to the number of classes as to obtain clusters that would correspond to the real classes. The euclidean distance was used as metric, and the heuristic to choose the initial cluster centroid positions was the k-means++ algorithm [10]. The implementation used was provided by the statistics and machine learning toolbox from Matlab. This implementation minimizes the sum, over all clusters, of the within-cluster sums of point-to-cluster-centroid.

Both the k-nn and the k-means can be used for single and multiclass problems without any adaptation. However, SVMs have to be generalized to multiclass problems. The simplest approach is to transform the multiclass problem into a binary problem. In this context, two variations are often considered: the one-against-all, and the one-against-one. In a one-against-all approach, 1 classifier per class is trained, where the class is considered positive, and all other classes are negatives. The output is then obtained by weighting the outputs of all classifiers. In one-against-one approaches, $c(c-1)/2$, with c the number of classes, classifiers are trained for each pair of classes, where the training samples belong to the two classes "being trained". The output is given as the majority of outputs from all the classifiers. The C-SVC implementation used in this work uses a one-versus-one approach.

To evaluate the classifiers, the area under the curve (auc) value [11], where the curve refers to the receiver operating characteristics (roc), was used. This metric is quite robust to unbalanced data, however, it is only applicable to the binary case. Nevertheless, generalizations have been proposed as to extend the auc to multiclass problems [12]. In this work, a simple one-against-all approach was taken, where the final auc value is taken as the average auc values obtained from considering, for each class, one class as positive and all others as negatives.

While the number of clusters passed as input to the k-means algorithm can define the number of classes, knowing which cluster corresponds to each class may not be easy. If the k-means would present good results, the class with the majority of points within a cluster, would tell us the class that the cluster corresponds. However, if bad results are obtained (which was the case, as presented in the following section), the cluster may not represent any class well. Thus, to evaluate the k-means, all combinations of cluster-class were tested, and only the best result was saved.

5 Experiments, Results and Analysis

The experimental procedure was separated into the binary problem and the multiclass problem. Nevertheless, and as aforementioned, each of these problems were tackled with the following steps: data preprocessing; feature extraction; feature selection; classifiers training and evaluation. However, not all methods described in the previous sections were used at the same time. In fact, 6 "dataset variants" were created for each problem, differing on the methods used for feature extraction/selection:

- Full dataset – no feature extraction, nor feature selection was used. This corresponds to the original dataset (501 features).
- PCA dataset – the dataset after extracting features from the original dataset using principal components analysis. Since this method is unsupervised, this dataset is equal to both the binary and the multiclass problem. 95% variance in the data was retained, resulting in 104 extracted features. Figure 2 presents the plot of the first 150 eigenvalues, where one can see that they are closed to 0 near the 100th eigenvalue;
- LLE dataset – the dataset after extracting features from the original dataset using locally linear embedding. The number of dimensions to retain was set to 104 (the same as the pca). Similarly to the pca dataset, this dataset is equal to both problems.
- Feature Corr dataset – the dataset after selecting features from the original dataset using the correlation between the features. The threshold used was 0.9, resulting in 138 features for both problems. **Important note:** I have mistakenly removed the features with lower correlation, instead of removing the features with higher correlation. Thus, the results with this dataset do not actually correspond to using the feature correlation extraction method.
- Class Corr dataset – the dataset after selecting features from the original dataset using the correlation between the features and each class. For the single class problem, the threshold was set to 0.45, since overall poor correlation with the class was found. This resulted in 71 selected features. However, the multiclass problem presented better correlation between the features and the classes. Therefore, the threshold was set to 0.75, resulting in 129 selected features.
- AUC dataset – the dataset after selecting features from the original dataset using the area under the (roc) curve value, where the features were used as as predictions and the class labels as targets. For the single class problem, an auc threshold of 0.85 was used, resulting in 80 selected features. For the multiclass problem, the auc values were 0.5 for each class. Thus, this dataset was not used for this latter problem.

For the single class problem, 43 features were common to all feature selection datasets (Feature Corr, Class Corr, AUC), whereas for the multiclass, 129 features were common to the Feature Corr and Class Corr datasets. Note that, since Class Corr selected exactly 129 features, all these features are also in the Feature Corr dataset.

For each of the dataset variants, and as presented in the previous section, three different classifiers were used for the classification process: the k-nearest neighbor (k-nn), support vector machines (SVM), and k-means clustering. Recall that k in the k-nn was set to 1, and k in the clustering was set to the number of classes (k=2 for the binary problem).

For the single class problem, both the k-nn and k-means were evaluated using k-fold cross validation with 10 folds. That is, for each fold (iteration) a different block of $\frac{1}{10}$ of the dataset is used for test and the remaining $\frac{9}{10}$ is used for training. For the SVM classifier, since training and evaluating, while tuning the parameters, takes a considerable amount of time, the leave-m-out cross-validation was used instead, where 70% of the dataset was used for training and 30% was used for test. The parameters were searched using a logarithmic base 2 search, where $C \in \{2^{-3}, 2^{-2}, \dots, 2^{3}\}$, and $\gamma \in \{2^{-5}, 2^{-3}, 2^{-1}, \dots, 2^{5}\}$.

Table 2 presents the obtained single class results for all the classifiers and all the dataset variations without balancing the class distribution. SVM avg_test_auc values are equal to max_test_auc, since they

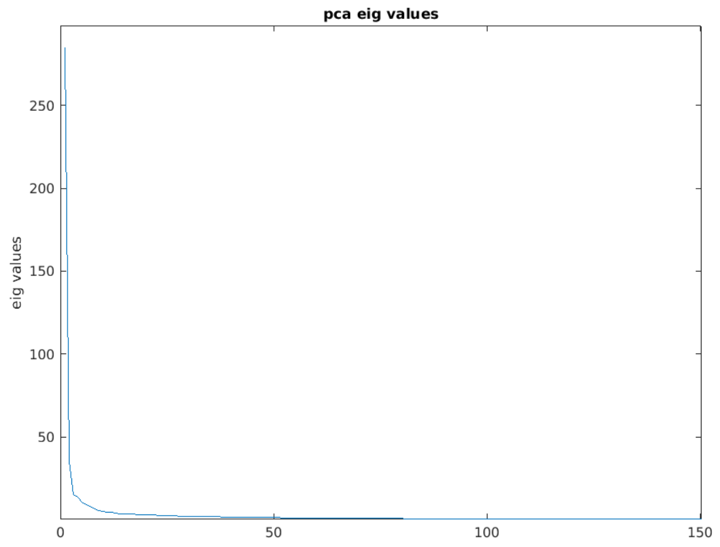


Figure 2: Plot of the first 150 eigenvalues, obtained with pca.

were trained with leave-m-out instead of k-fold cross validation, thus only 1 iteration occurred. k-means avg_train_train_auc is empty since there is not a well defined train/test in clustering analysis. It is observable from table 2 that the best results are obtained with the 1-NN. However, SVM also present good results for the feature selection datasets, specially for Class Corr and AUC, where the average test auc values are over 0.97. However, for the first two dataset variants, it seems that SVMs produced some over-fitting. This is evident by the fact that the average train auc values are 1 and the test auc values are close to 0.5 or 0.6. The following output log evidences this over-fitting, by presenting the number of support vectors (SVs) for each of the datasets:

```
Number of SV for each dataset:  (dataset training size  7209)
Full Dataset (C=2, γ=0.03125):  6485 (89.957% of the training data)
PCA (C=2, γ=0.03125):  6037 (83.7425% of the training data)
LLE (C=2, γ=0.03125):  2793 (38.7432% of the training data)
Feature Corr (C=2, γ=0.03125):  4654 (64.5582% of the training data)
Class Corr (C=4, γ=0.03125):  878 (12.1792% of the training data)
AUC (C=16, γ=0.03125):  1145 (15.8829% of the training data)
```

The closer is the number of SVs to the dataset training size, the higher the probability of over-fitting, since the generalization is highly degraded.

The k-means classifier produced the worse results for the single class problem (table 2). This was to be expected, since this is an unsupervised learning approach. However, the maximum auc value in all datasets, except for the LLE, is higher than 0.8. This fact leads one to assume that good results could have been obtained. Maybe using a different distance metric, or a different heuristic for the choice of the initial centroids.

For the multi class problem, 5 folds were used instead, and the parameter tunning for the SVM classifier used 75% of the dataset for training and 25% for test. Again, the parameters were searched using a logarithmic base 2 search, where $C \in \{2^{-2}, 2^{0}, \ldots, 2^{12}\}$, and $\gamma \in \{2^{-8}, 2^{-6}, \ldots, 2^{2}\}$.

		1NN	**SVM**	**k-means (k=2)**
Full Dataset	avg_train_auc	1	1	
	avg_test_auc	0.99599	0.53295	0.5642
	max_test_auc	0.99942	0.53295	0.8258
PCA	avg_train_auc	1	1	
	avg_test_auc	0.99587	0.60271	0.632
	max_test_auc	0.99942	0.60271	0.8409
LLE	avg_train_auc	1	0.9971	
	avg_test_auc	0.97517	0.89263	0.5013
	max_test_auc	0.98487	0.89263	0.5046
Feature Corr	avg_train_auc	1	1	
	avg_test_auc	0.98161	0.84128	0.6857
	max_test_auc	0.98603	0.84128	0.8209
Class Corr	avg_train_auc	1	0.97238	
	avg_test_auc	0.9696	0.97402	0.6313
	max_test_auc	0.98137	0.97402	0.838
AUC	avg_train_auc	1	0.99345	
	avg_test_auc	0.95986	0.97208	0.6046
	max_test_auc	0.96973	0.97208	0.8532

Table 2: Single class results using unbalanced datasets.

Tables 3 and 4 present the results for the multiclass problem using unbalanced datasets and balanced datasets, respectively. Appendices 7 and 8 present the results for each of the classes. It seems that there was not a big difference between the results using balanced or unbalanced dataset. The main reason is that even with the unbalanced datasets, good overall results were obtained, with the exception of the results for the k-means. However, cluster analysis has no consideration for the class, thus balancing the data may not necessarily lead to better the results.

An interesting observation is that the SVM was considerably better for the multiclass problem than for the single class problem. Even for the case of the full dataset, good generalization was achieved. In fact, the number of SVs has considerably decreased, as can be seen in the following log:

```
Number of SV for each dataset:  (dataset training size  6749)
Full Dataset (C=64, g=0.0039062):  2750 (40.7468% of the training data)
PCA (C=64, γ=0.0039062):  2451 (36.3165% of the training data)
LLE (C=16, γ=0.0039062):  2728 (40.4208% of the training data)
Feature Corr (C=64, γ=0.0039062):  2616 (38.7613% of the training data)
Class Corr (C=4096, γ=0.015625):  1703 (25.2334% of the training data)
```

Looking at the previous log, it seems that higher values of C produced better results. It is possible that the SVM for the binary problem was not well tuned, since the search was done in a more limited range.

Differently from the binary problem, in the multiclass problem, no significant difference between the results using different feature selection/extraction was observed. This suggests that the features in the original dataset are already sufficiently discriminatory for this problem. However, in terms of execution times (not presented in this report), using the full dataset is computationally demanding, specially for training the SVMs. In the context of execution time, k-nn (with k=1) was the fastest of the classifiers.

		k-NN	SVM	k-means (k=nr_classes)
Full Dataset	avg_train_auc	1	1	
	avg_test_auc	0.98048	0.9859	0.53351
	max_test_auc	0.98287	0.9859	0.61596
PCA	avg_train_auc	1	0.99987	
	avg_test_auc	0.97631	0.98735	0.52109
	max_test_auc	0.97782	0.98735	0.55141
LLE	avg_train_auc	1	0.98302	
	avg_test_auc	0.9331	0.94879	0.50151
	max_test_auc	0.93889	0.94879	0.59308
Feature Corr	avg_train_auc	1	0.99966	
	avg_test_auc	0.94744	0.9869	0.50941
	max_test_auc	0.95343	0.9869	0.60907
Class Corr	avg_train_auc	1	0.99286	
	avg_test_auc	0.92882	0.95818	0.55268
	max_test_auc	0.93155	0.95818	0.60981

Table 3: Multiclass results using unbalanced datasets.

		k-NN	SVM	k-means (k=nr_classes)
Full Dataset	avg_train_auc	1	1	
	avg_test_auc	0.98058	0.99005	0.52682
	max_test_auc	0.98293	0.99005	0.57367
PCA	avg_train_auc	1	1	
	avg_test_auc	0.97575	0.9872	0.51201
	max_test_auc	0.9815	0.9872	0.56726
LLE	avg_train_auc	1	0.98373	
	avg_test_auc	0.94132	0.96053	0.53145
	max_test_auc	0.94703	0.96053	0.60002
Feature Corr	avg_train_auc	1	1	
	avg_test_auc	0.9505	0.98648	0.45295
	max_test_auc	0.9605	0.98648	0.52614
Class Corr	avg_train_auc	1	0.98133	
	avg_test_auc	0.93051	0.95841	0.57559
	max_test_auc	0.93347	0.95841	0.6153

Table 4: Multiclass results using balanced datasets.

6 Conclusion

This work has tackle human activity recognition by developing efficient activity classifiers. Two different scenarios were considered: a binary problem (walking vs. non-walking) and a multiclass problem. Each of these problems was solved using a similar set of steps: data preprocessing; feature extraction; feature selection; classifiers training and evaluation. Depending on the method of feature selection or feature extraction, variants of the original dataset were created, differing on the considered features. For the classification process, the classifiers k-nearest neighbor, support vector machines and k-means were used.

Results showed that both the k-nn and the SVMs have good performance under both considered scenarios, while the k-means as given poor results. This was to be expected, due to the unsupervised approach taken by cluster analysis. In terms of execution time, the k-nn (with k=1) was significantly faster than k-means and SVMs. In fact, SVMs took considerable amounts of time to be trained.

As a final note, a few considerations could have been implemented to obtain better results:

- A wider range for the grid-search of SVM parameters for the binary problem could have been performed. This could have enhanced the results of this classifier;
- Could have used different distance metrics for the k-means cluster analysis. However, the euclidean distance seemed to work well for the k-nn;
- Both used feature extraction methods were unsupervised. This means that a lot of information related with the true label is not considered in the process of extracting features;
- Feature selection methods could also have been performed after the feature extraction methods. This could have enhanced the impact of each feature, and also, further reduce the dataset dimensionality;
- More classifiers could have been used. Specially supervised approaches. These could have been used for further comparisons.

Nonetheless, the overall results were quite satisfactory, and thus, the objective of this work was successfully completed.

7 Appendix A

The following tables present the multiclass problem average test results for each of the classes using unbalanced datasets.

Table 5: Average test auc per class for classifier k-NN

	C1	C2	C3	C4	C5	C6
Full Dataset	0.99757	0.94733	0.9555	0.99669	0.99019	0.99564
PCA	0.99553	0.93717	0.94699	0.99564	0.98789	0.99465
LLE	0.97317	0.87951	0.90232	0.97487	0.90708	0.96164
Feature Corr	0.97957	0.92261	0.93984	0.97847	0.90263	0.96154
Class Corr	0.8776	0.84391	0.87168	0.99692	0.98978	0.99303

Table 6: Average test auc per class for classifier SVM

	C1	C2	C3	C4	C5	C6
Full Dataset	0.99486	0.97336	0.97674	0.98692	0.99325	0.99029
PCA	0.99743	0.97219	0.97221	0.99419	0.99485	0.99324
LLE	0.99183	0.90852	0.92593	0.95609	0.95807	0.95231
Feature Corr	1	0.96573	0.97275	0.99651	0.99449	0.99191
Class Corr	0.93082	0.91544	0.92176	0.99855	0.9901	0.99239

Table 7: Average test auc per class for classifier k-means (k=nr_classes)

	C1	C2	C3	C4	C5	C6
Full Dataset	0.57243	0.50976	0.55322	0.49356	0.50139	0.57071
PCA	0.53391	0.54205	0.4756	0.60469	0.47522	0.49509
LLE	0.54968	0.46943	0.4886	0.52281	0.50308	0.47547
Feature Corr	0.444	0.51766	0.43096	0.56386	0.50178	0.59819
Class Corr	0.61239	0.46217	0.58091	0.57443	0.56377	0.52242

8 Appendix B

The following tables present the multiclass problem average test results for each of the classes using balanced datasets.

Table 8: Average test auc per class for classifier k-NN

	C1	C2	C3	C4	C5	C6
Full Dataset	0.99573	0.94801	0.96003	0.99637	0.99032	0.99303
PCA	0.99566	0.93635	0.94658	0.99652	0.9872	0.99218
LLE	0.96451	0.89366	0.9229	0.97909	0.92402	0.96373
Feature Corr	0.97475	0.92484	0.93357	0.98258	0.91871	0.96856
Class Corr	0.88142	0.85255	0.8663	0.99588	0.99288	0.99403

Table 9: Average test auc per class for classifier SVM

	C1	C2	C3	C4	C5	C6
Full Dataset	0.99255	0.97936	0.98328	0.99291	0.99574	0.99644
PCA	0.99291	0.97225	0.9726	0.99291	0.99609	0.99644
LLE	0.9968	0.92528	0.95162	0.96098	0.96656	0.96193
Feature Corr	0.99964	0.96976	0.96015	0.99433	0.99715	0.99787
Class Corr	0.93117	0.90856	0.91924	0.99752	0.99822	0.99573

Table 10: Average test auc per class for classifier k-means (k=nr_classes)

	C1	C2	C3	C4	C5	C6
Full Dataset	0.44026	0.57845	0.51561	0.54715	0.57021	0.50927
PCA	0.45937	0.5261	0.49648	0.5763	0.52353	0.49026
LLE	0.46584	0.56937	0.57598	0.43297	0.5668	0.57775
Feature Corr	0.48609	0.4242	0.46717	0.50977	0.40528	0.42521
Class Corr	0.71302	0.4441	0.53277	0.49859	0.62913	0.63593

References

[1] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, “A public domain dataset for human activity recognition using smartphones.” in *ESANN*, 2013.

[2] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques.* Elsevier, 2012.

[3] S. Kotsiantis, D. Kanellopoulos, P. Pintelas *et al.*, “Handling imbalanced datasets: A review,” *GESTS International Transactions on Computer Science and Engineering*, vol. 30, no. 1, pp. 25–36, 2006.

[4] A. K. Jain, R. P. W. Duin, and J. Mao, “Statistical pattern recognition: A review,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 1, pp. 4–37, 2000.

[5] V. Franc and V. Hlavác, “Statistical pattern recognition toolbox for matlab,” *Prague, Czech: Center for Machine Perception, Czech Technical University*, 2004.

[6] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification.* John Wiley & Sons, 2012.

[7] C.-C. Chang and C.-J. Lin, “Libsvm: a library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[8] C.-W. Hsu, C.-C. Chang, C.-J. Lin *et al.*, “A practical guide to support vector classification,” 2003.

[9] J. MacQueen *et al.*, “Some methods for classification and analysis of multivariate observations,” in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA., 1967, pp. 281–297.

[10] D. Arthur and S. Vassilvitskii, “k-means++: The advantages of careful seeding,” in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms.* Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[11] T. Fawcett, “An introduction to roc analysis,” *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[12] D. J. Hand and R. J. Till, “A simple generalisation of the area under the roc curve for multiple class classification problems,” *Machine learning*, vol. 45, no. 2, pp. 171–186, 2001.